1. (Previously Presented) A processor-readable storage medium comprising processor-executable instructions to be executed by one or more processors to perform a method comprising:

constructing a filter graph to process a data stream from a source file which is received by a media player;

processing the data stream through the filter graph;

receiving an instruction from a user to load a new filter into the filter graph, wherein the new filter is new to the media player and modifies data of the data stream;

registering the new filter with the media player when the new filter is installed, such that the new filter can be recognized by the media player for loading based on registration parameters stored in a registry, and such that the new filter cannot subsequently be pirated for use on other types of media players;

recognizing the new filter based on the registration parameters stored in the registry;

dynamically loading the new filter into the filter graph during the processing of the data stream through the filter graph in response to the instruction received from the user;

after dynamically loading the new filter into the filter graph, processing the data stream through the filter graph which includes the new filter; and

communicating information represented by the data stream to the user via the media player.

2. (Previously Presented) A processor-readable storage medium as recited in claim 1, wherein the constructing a filter graph comprises:

reading a registry of filter characteristics;

identifying a plurality of filters available for operation in a filter graph based on the filter characteristics;

creating from the plurality of filters, an instance of a class of filters appropriate for rendering the data stream, each filter in the class of filters operative to conduct a processing operation and having at least one input pin and at least one output pin; and

connecting the pins of the filters in the class of filters to assemble the filter graph, the filter graph comprising connected filters wherein the first filter in the filter graph accepts the data stream and the final filter in the filter graph renders the data stream.

3. (Previously Presented) A processor-readable storage medium as recited in claim 1, wherein the dynamically loading comprises:

automatically stopping the processing at a current location in the data stream;

automatically loading the new filter into the filter graph; and

automatically restarting the processing at the current location in the data stream, the processing after the restarting including processing the data stream through the new filter.

4. (Previously Presented) A processor-readable storage medium as recited in claim 3, wherein the dynamically loading the new filter comprises:

deconstructing the filter graph at a connection point between two filters;

inserting the new filter at the connection point; and

reconstructing the filter graph such that it includes the new filter inserted between the two filters.

5. (Previously Presented) A processor-readable storage medium comprising processor-executable instructions to be executed by one or more processors to perform a method comprising:

constructing a filter graph to process a data stream from a source file which is received by a media player;

processing the data stream through the filter graph;

receiving an instruction from a user to load a new filter into the filter graph, wherein the new filter modifies the data of the data stream;

receiving a call to register a plug-in when the new filter is installed, such that the new filter can be recognized by the media player for loading based on registration parameters stored in a registry, and such that the new filter cannot subsequently be pirated for use on other types of media players; and

in accordance with the call, receiving a set of registration parameters comprising:

a pwszFriendlyName parameter designating a name for the plug-in;

a pwszDescription parameter designating a description of the plug-in;

a pwszUninstallString parameter designating an uninstall string for uninstalling the plug-in;

a dwPriority parameter designating an integer value containing a priority position of the plug-in in a chain of currently enabled plug-ins;

a guidPluginType parameter designating a globally unique identifier that specifies a type for the plug-in;

a Clsid parameter designating a class identifier of the plug-in;

a cMediaType parameter designating a count of media types supported by the plug-in; and

a pMediaType parameter designating a pointer to an array of media types that enumerates supported media types for the plug-in;

registering the new filter with the media player using the set of registration parameters;

recognizing the new filter based on the registration parameters stored in the registry;

dynamically loading the new filter into the filter graph during the processing of the data stream through the filter graph in response to the instruction received from the user;

after dynamically loading the new filter into the filter graph, processing the data stream through the filter graph which includes the new filter; and

communicating information represented by the data stream to the user via the media player.

6. (Previously Presented) A processor-readable storage medium as recited in claim 5, comprising further processor-executable instructions configured for storing the set of registration parameters according to a specific format in a registry of an operating system on a machine wide basis.

7. (Previously Presented) A processor-readable storage medium as recited in claim 6, wherein the specific format comprises:

a plug-in type specified by a guidPluginType parameter;

a plug-in major format specified by a pMediaType parameter that further specifies the plug-in type;

a plug-in minor format specified by another pMediaType parameter that further specifies the plug-in major format;

a unique identification of a plug-in specified by a Clsid parameter, the unique identification identifying a plug-in capable of processing the plug-in minor format;

a configuration heading for the plug-in type that includes the unique identification of the plug-in specified by the Clsid parameter;

a description of the plug-in type specified by the pwszDescription parameter;

a name of the plug-in type specified by the pwszFriendlyName parameter;

a priority of the plug-in type specified by the dwPriority parameter; and

an uninstall path specified by the pwszUninstallString parameter.

8. (Previously Presented) A processor-readable storage medium as recited in claim 5, comprising further processor-executable instructions configured for storing a subset of the set of registration parameters according to a specific format in a registry of an operating system on a per user basis.

9. (Previously Presented) A processor-readable storage medium as recited in claim 8, wherein the specific format comprises:

a configuration heading for the plug-in type that includes the unique identification of the plug-in specified by the Clsid parameter;

an enable indicator permitting the user to enable the plug-in; and

a priority of the plug-in type specified by the dwPriority parameter.

10. (Previously Presented) A processor-readable storage medium as recited in claim 5, comprising further processor-executable instructions configured for:

constructing a filter graph to process a data stream from a source file;

processing the data stream through the filter graph;

receiving an instruction to load the plug-in;

searching the registry for the plug-in;

recognizing the plug-in based on the set of registration parameters stored in the registry; and

dynamically loading the plug-in into the filter graph during the processing.

11. (Previously Presented) A processor-readable storage medium as recited in claim 10, wherein the dynamically loading comprises:

automatically stopping the processing at a current location in the data stream;

automatically loading the plug-in into the filter graph; and

automatically restarting the processing at the current location in the data stream, the processing after the restarting including processing the data stream through the plug-in.

12. (Previously Presented) A processor-readable storage medium as recited in claim 11, wherein the automatically loading the plug-in comprises:

deconstructing the filter graph at a connection point between two filters;

inserting the plug-in at the connection point; and

reconstructing the filter graph such that it includes the plug-in inserted between the two filters.

13. (Previously Presented) A processor-readable storage medium as recited in claim 10, wherein the constructing a filter graph comprises:

reading a table of filter characteristics;

identifying a plurality of filters available for operation in a filter graph based on the reading;

creating from the plurality of filters, an instance of a class of filters appropriate for rendering the data stream, each filter in the class of filters operative to conduct a processing operation and having at least one input pin and at least one output pin; and

connecting the pins of the filters in the class of filters to assemble the filter graph, the filter graph comprising connected filters wherein the first filter in the filter graph accepts the data stream and the final filter in the filter graph renders the data stream.

14. (Previously Presented) A processor-readable storage medium as recited in claim 5 having stored thereon a data structure providing a format for storing the registration parameters, the data structure comprising:

a PLUG-IN_TYPE field configured to contain a registration parameter;

a PLUG-IN_MAJOR_FORMAT field configured to contain a registration parameter;

a PLUG-IN_MINOR_FORMAT field configured to contain a registration parameter;

a PLUG-IN_TYPE_CONFIGS field configured to contain a registration parameter;

a PLUG-IN_ID field configured to contain a registration parameter;

a DESCRIPTION field configured to contain a registration parameter;

a NAME field configured to contain a registration parameter;

a PRIORITY field configured to contain a registration parameter; and

a UNINSTALLPATH field configured to contain a registration parameter.

15. (Previously Presented) A processor-readable storage medium as recited in claim 5 having stored thereon a data structure providing a format for storing the registration parameters, the data structure comprising:

a PLUG-IN_TYPE_CONFIGS field configured to contain a registration parameter;

a PLUG-IN_ID field configured to contain a registration parameter;

an ENABLED field configured to contain a registration parameter; and

a PRIORITY field configured to contain a registration parameter.

16. (Canceled)

17. (Canceled)

18. (Canceled)

19. (Canceled)

20. (Canceled)

21. (Previously Presented) A method comprising:

receiving streaming data at a media player;

constructing a filter graph based on a data type of the streaming data received at the media player, wherein the streaming data is in a format know to the media player;

processing the streaming data through the filter graph;

receiving an instruction from a user to load a new filter into the filter graph, wherein the new filter is new to the media player and modifies the streaming data;

registering the new filter with the media player when the new filter is installed, such that the new filter can be recognized by the media player for loading based on registration parameters stored in a registry, and such that the new filter cannot subsequently be pirated for use on other types of media players;

recognizing the new filter based on the registration parameters stored in the registry;

dynamically loading the new filter into the filter graph during the processing of the streaming data through the filter graph in response to the instruction received from the user;

after dynamically loading the new filter into the filter graph, processing the data stream through the filter graph which includes the new filter; and

communicating information represented by the data stream to the user via the media player.

22. (Original) A method as recited in claim 21, wherein the constructing a filter graph comprises:

reading a registry of filter characteristics;

identifying filters available to process the streaming data based on the filter characteristics;

creating from the filters, an instance of a class of filters appropriate for rendering the streaming data, each filter in the class of filters operative to conduct a processing operation and having at least one input pin and at least one output pin; and

connecting the pins of the filters in the class of filters to assemble the filter graph, the filter graph comprising connected filters wherein the first filter in the filter graph accepts the streaming data and the final filter in the filter graph renders the streaming data.

23. (Original) A method as recited in claim 21, wherein the dynamically loading comprises:

stopping the processing at a current location in the streaming data;

deconstructing the filter graph at a connection point between two filters;

inserting the new filter at the connection point;

reconstructing the filter graph such that it includes the new filter inserted between the two filters; and

restarting the processing at the current location in the data stream, the processing after the restarting including processing the data stream through the new filter.

24. (Original) A method as recited in claim 23, wherein the stopping, the deconstructing, the inserting, the reconstructing and the restarting are performed automatically by a media player.

25. (Previously Presented) A computer comprising:

one or more processors;

a registration function which executes on the one or more processors to receive a set of registration parameters from a filter plug-in which has been selected by a user for use in processing a data stream, the registration parameters comprising:

a pwszFriendlyName parameter designating a name for the plug-in;

a pwszDescription parameter designating a description of the plug-in;

a pwszUninstallString parameter designating an uninstall string for uninstalling the plug-in;

a dwPriority parameter designating an integer value containing a priority position of the plug-in in a chain of currently enabled plug-ins;

a guidPluginType parameter designating a globally unique identifier that specifies a type for the plug-in;

a Clsid parameter designating a class identifier of the plug-in;

a cMediaType parameter designating a count of media types supported by the plug-in; and

a pMediaType parameter designating a pointer to an array of media types that enumerates supported media types for the plug-in.

26. (Original) A computer as recited in claim 25, further comprising a registry that includes the set of registration parameters configured on a machine wide basis according to the following format:

a plug-in type specified by a guidPluginType parameter;

a plug-in major format specified by a pMediaType parameter that further specifies the plug-in type;

a plug-in minor format specified by another pMediaType parameter that further specifies the plug-in major format;

a unique identification of a plug-in specified by a Clsid parameter, the unique identification identifying a plug-in capable of processing the plug-in minor format;

a configuration heading for the plug-in type that includes the unique identification of the plug-in specified by the Clsid parameter;

a description of the plug-in type specified by the pwszDescription parameter;

a name of the plug-in type specified by the pwszFriendlyName parameter;

a priority of the plug-in type specified by the dwPriority parameter; and

an uninstall path specified by the pwszUninstallString parameter.

27. (Original) A computer as recited in claim 25, further comprising a registry that includes a subset of the set of registration parameters configured on a per user basis according to the following format:

a configuration heading for the plug-in type that includes the unique identification of the plug-in specified by the Clsid parameter;

an enable indicator permitting the user to enable the plug-in; and

a priority of the plug-in type specified by the dwPriority parameter.

28. (Original) A computer as recited in claim 25, further comprising a media player configured to register a filter plug-in according to the registration parameters.

29. (Original) A computer as recited in claim 28, further comprising a filter graph manager configured to construct a filter graph based the registration parameters and on a data type of a data stream received by the media player.

30. (Original) A computer as recited in claim 29, further comprising a dynamic plug-in loader configured to automatically stop the filter graph from processing the data stream, determine an enabled filter plug-in based on the registration parameters, deconstruct the filter graph, insert the enabled filter plug-in into the filter graph, and restart the processing of the data stream.

31. (Previously Presented) A computer comprising:

means for constructing a filter graph to process a data stream from a source file which is received by a media player;

means for processing the data stream through the filter graph;

means for receiving an instruction from a user to load a new filter into the filter graph, wherein the new filter is new to the media player and modifies the data of the data stream;

means for registering the new filter with the media player when the new filter is installed, such that the new filter can be recognized by the media player for loading based on registration parameters stored in a registry, and such that the new filter can not subsequently be pirated for use on other types of media players;

means for recognizing the new filter based on the registration parameters stored in the registry; and

means for dynamically loading the new filter into the filter graph during the processing of the data stream through the filter graph in response to the instruction received from the user;

means for processing the data stream through the filter graph which includes the new filter; and

means for communicating information represented by the data stream to the user via the media player.

32. (Original) A computer as recited in claim 31, wherein the means for constructing a filter graph comprises:

means for reading a registry of filter characteristics;

means for identifying a plurality of filters available for operation in a filter graph based on the filter characteristics;

means for creating from the plurality of filters, an instance of a class of filters appropriate for rendering the data stream, each filter in the class of filters operative to conduct a processing operation and having at least one input pin and at least one output pin; and

means for connecting the pins of the filters in the class of filters to assemble the filter graph, the filter graph comprising connected filters wherein the first filter in the filter graph accepts the data stream and the final filter in the filter graph renders the data stream.

33. (Original) A computer as recited in claim 31, wherein the means for dynamically loading comprises:

means for automatically stopping the processing at a current location in the data stream;

means for automatically loading the new filter into the filter graph; and

means for automatically restarting the processing at the current location in the data stream, the processing after the restarting including processing the data stream through the new filter.

34. (Original) A computer as recited in claim 33, wherein the means for automatically loading the new filter comprises:

means for deconstructing the filter graph at a connection point between two filters;

means for inserting the new filter at the connection point; and

means for reconstructing the filter graph such that it includes the new filter inserted between the two filters.

35. (Canceled)

36. (Canceled)

37. (Canceled)